

RL-TR-96-108
Final Technical Report
May 1996



AUTOMATIC EXTRACTION OF DRAINAGE NETWORK FROM DIGITAL TERRAIN ELEVATION DATA (DTED)

University of Maine

**Mohamad T. Musavi, Ph.D., Greg Hawkins, Alan Fern,
and Curt Ruffing**

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

19960730 082

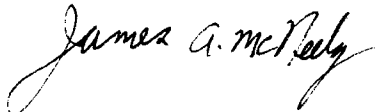
**Rome Laboratory
Air Force Materiel Command
Rome, New York**

DTIC QUALITY INSPECTED 1

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be releasable to the general public, including foreign nations.

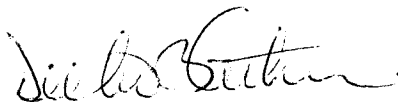
RL-TR-96-108 has been reviewed and is approved for publication.

APPROVED:



JAMES A. MCNEELY
Project Engineer

FOR THE COMMANDER:



DELBERT B. ATKINSON, Colonel, USAF
Director of Intelligence & Reconnaissance

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify Rome Laboratory/ (IRRP), Rome NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave Blank)		2. REPORT DATE May 1996		3. REPORT TYPE AND DATES COVERED Final ----	
4. TITLE AND SUBTITLE AUTOMATIC EXTRACTION OF DRAINAGE NETWORK FROM DIGITAL TERRAIN ELEVATION DATA (DTED)				5. FUNDING NUMBERS C - F30602-95-C-0011 PE - 63260R PR - 3479 TA - 17 WU - PA	
6. AUTHOR(S) Mohamad T. Musavi, Ph.D., Greg Hawkins, Alan Fern, and Curt Ruffing				8. PERFORMING ORGANIZATION REPORT NUMBER N/A	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of Maine Department of Electrical & Computer Engineering 5708 Barrows Hall Orono ME 04469-5708				10. SPONSORING/MONITORING AGENCY REPORT NUMBER RL-TR-96-108	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Rome Laboratory/IRRP 32 Hangar Rd Rome NY 13441-4114					
11. SUPPLEMENTARY NOTES Rome Laboratory Project Engineer: James A. McNeely/IRRP/(315) 330-2110					
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.				12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) This report describes an approach to the automatic extraction of a drainage network from Digital Terrain Elevation Data (DTED). The extraction of drainage networks from topological data is one of the more important uses of such information. An extracted network may be used in many applications, such as determining drainage network metrics or selecting control points for image registration and mapping applications. Neural networks have been shown to possess generalization properties that make them well suited to this problem, which calls for a generalized solution. In this research, a biologically inspired network has been designed and applied to automatic extraction of drainage networks. Several DTED test files were used for testing of the developed model. The results indicate the success of neural networks in the extraction of drainage networks from DTED.					
14. SUBJECT TERMS Automatic feature extraction, Neural network architecture, Mathematic model for Dnet and drainage network				15. NUMBER OF PAGES 40	
				16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED		18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED		19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	
				20. LIMITATION OF ABSTRACT UL	

Acknowledgment

This work was made possible by a contract (F30602-95-C-0011) from Rome Laboratory. The technical contact person at Rome Laboratory is Mr. James McNeely.

We would like to thank Rome Laboratory for providing the opportunity to work on this project and Mr. McNeely for providing DTED, imaging data, guidance, and technical support.

In addition to the Principal Investigator, Mohamad Musavi, the work involved one graduate student, Mr. Greg Hawkins, and two undergraduate students, Mr. Alan Fern and Curt Ruffing, from the Department of Electrical and Computer Engineering at the University of Maine.

Table of Contents

ACKNOWLEDGEMENT.....	i
TABLE OF CONTENTS	ii
1. INTRODUCTION	1-1
2. NEURAL NETWORK ARCHITECTURE	2-1
2.1 CONCEPTUAL APPROACH.....	2-1
2.2 MATHEMATICAL MODEL.....	2-3
2.2.1 Constraints	2-3
2.2.2 Indexing	2-4
2.2.3 Model for Property 1	2-5
2.2.4 Model for Property 2.....	2-6
2.2.5 Model for Property 3.....	2-6
2.2.6 Dnet Model.....	2-7
3. EXPERIMENTS	3-1
3.1 TESTING DATA.....	3-1
3.2 RESULTS	3-3
3.2.1 Test Region 1	3-3
3.2.2 Test Region 2.....	3-4
3.3 DISCUSSION OF PARAMETERS	3-7
3.3.1 S_{max}	3-7
3.3.2 α_{pi}	3-7
3.3.3 N_{max}	3-7
3.3.4 σ_{pi}	3-8
3.3.5 C	3-8
3.3.6 C_{max}	3-8
3.3.7 β	3-8
3.3.8 D	3-9
3.3.9 Y_o	3-9
3.3.10 H_{cut}	3-9
4. CONCLUSIONS	4-1
5. IMPLEMENTATION	5-1
5.1 INPUT.....	5-1
5.2 OUTPUT	5-1
5.3 HEIGHT	5-2
5.4 WIDTH	5-2
5.5 ITER	5-2
5.6 SAVEINT	5-2
5.7 OPTIONAL ARGUMENTS	5-2
6. REFERENCES.....	6-1

1. Introduction

Extraction of drainage networks from digital elevation data has many useful applications. Hard to measure drainage system metrics such as, channel length, gradient, and sinuosity may be easily and accurately computed using a drainage network mapped from elevation data. Also since the location of an elevation data point is known to a specified accuracy, an extracted drainage network may be used in the selection of ground reference points. These points may then be used in various mapping applications as well as registration of aerial images. The traditional approach to automate drainage network extraction has been to digitize drainage segments as seen on topological maps. This approach gives less than adequate results in many cases. The accuracy of the results are limited by the inconsistencies of cartographer interpretations. Also, if information about small stream segments is desired, cartographic maps may not provide the desired accuracy. A better solution would be to use Digital Terrain Elevation Data (DTED) to determine a drainage network. Such a solution would remedy most of the shortcomings involved with the cartographic method.

DTED is one of the richest resources of topological information. It has been gathered for much of the earth including the entire continental United States and is becoming increasingly more assessable via the Internet, CD-ROM, and magnetic tape. A DTED data base consist of many files. Each file corresponds to a cell at a specific longitude and latitude and stores a matrix of the earth's elevation values (referenced to sea level) within that cell. The most widely available DTED files, and the ones used in this study, are produced by the Defense Mapping Agency (DMA) and have a cell size of 1 degree by 1 degree with a spatial resolution of 1:250,000. Another more accurate format, distributed by both the DMA and the U.S. Geological Survey (USGS) has a cell size of 7.5 minutes by 7.5 minutes with a spatial resolution of 1:24,000.

Essentially the problem of drainage network extraction from elevation data is to find points in the DTED files which meet the following conditions:

- A point must be a member of a valid valley segment (river, stream, etc.) or a larger drainage basin (lake, pond, etc.).
- The elevations along a valid valley segment must decrease in one direction (flow direction) since water flows from higher to lower elevations.
- A valid valley segment must have a source where water can enter into and travel in the flow direction of the particular segment. A source may originate either at the junction of DTED cells, another valley segment, or a point at the end of the segment whose elevation is the highest in the segment.
- A valid valley segment must have a destination for it's flow of water. In other words the water flow cannot stop abruptly. A destination can be another valley segment, a larger drainage basin, or the junction of other DTED cells.

Implementation of a drainage network extraction technique is complicated by errors existing in the DTED. These errors are a result of and vary with the topographic sampling and digitization techniques used to generate the data. Common errors are the inclusion of artificial pits and ridge points. Also narrow streams may appear to have discontinuities depending on the resolution of the DTED. This makes finding a connected drainage network a difficult task. A solution to the drainage network extraction problem must consider such sources of error. Lee, Snyder, and Fisher [1] offer a more complete discussion of the effects of DTED errors on feature extraction.

Several notable attempts have been made to automate the extraction of drainage networks from elevation data. O'Calaghan and Mark [2] used a technique where a drainage direction was assigned to each pixel. An iterative computation was then performed in which drainage accumulation values were updated for each pixel based on a weighted sum of the accumulation values of surrounding pixels. A pixel was labeled as being part of the drainage network if its accumulation value was above a specified threshold. Jenson [3] used a moving 3x3 pixel window to label possible drainage points by searching for local minima between two non-adjacent pixels. Localized rules were then used to extract a possible drainage network based on a user-specified distance and elevation threshold. These techniques suffer from several deficiencies which are linked to the local nature of the algorithms. The global reasoning necessary to establish links between separated stream segments is not achievable by these techniques. This causes for the extracted network to be broken into unconnected segments. Also these algorithms may not be able to weed out local minima in the DTED which are not part of the overall drainage network.

DNESYS, offered by Qian, Ehrich, and Campbell [4], is an attempt to overcome the above limitations. It is an expert system based method, which uses both local operators and global reasoning to solve for a valid drainage network. First, using a local operator and a reasoning process, groups of pixels which represent stream segments are labeled and then given to a hypothesis generator. The hypothesis generator suggest links between spatially related segments and decides which segments are not part of the overall drainage network. This more global approach at drainage network extraction produces better results than the local algorithms described.

The purpose of this project is to investigate a neural network approach for the extraction of drainage networks. Neural networks have been used to solve a wide variety of problems. As their applicability becomes increasingly apparent, more researchers are taking interest in this still relatively new field. Our task is to further demonstrate the usefulness of neural networks by solving a real world problem. Much like the human auditory system, which is capable of distinguishing a voice among a disarray of sound waves, we would like a neural network to distinguish a drainage network from the midst of a DTED file. There are three main advantages in using a neural network approach. First, many real world problems require fast solutions, sometimes so fast that dedicated hardware must be built to carry out a particular algorithm. A neural network solution to the drainage network extraction problem would be far simpler to implement in hardware than a rule-based solution using an expert system. This is because neural networks are constructed from large quantities of simple parts, as opposed to a small number of complicated rule-based engines, making them very well suited for VLSI implementation.

Second, neural networks have been shown to be more tolerant to noise than many conventional algorithms applied to the same task. This is because neural networks distribute the problem among many neurons. A change in a small percentage of the neurons, occurring because of moderate noise, does not have a large impact on the overall network. This makes them extremely useful tools when applied to noisy data such as DTED. Third, neural networks are extremely general, in that the same network may be used to solve many different problems. Because of this, any neural network methods developed to solve the drainage network problem will most likely find use in other applications.

Section 2 of this report provides the mathematical foundation for the developed neural network and Section 3 gives the results of our experiments on two DTED samples. Section 4 provides a conclusion on the methodology and Section 5 is a guide for compiling and running the software.

2. Neural Network Architecture

This section will provide a description of a drainage neural network (Dnet) architecture for finding the drainage regions within elevation data. First, the conceptual approach to Dnet's design will be described, this is intended to give the reader an intuitive feel for the network rather than viewing it as a purely mathematical structure. Second, the mathematical implementation will be discussed which carries out the network described in the conceptual approach section.

2.1 Conceptual Approach

When considering a neural network solution to the drainage network extraction problem, attention must be given to the fact that this problem is both *local* and *global* in nature. If a neural network was given only a local view of the data by showing it an $N \times N$ window of elevation values, the network would have no knowledge of what surrounded the narrow scope of its viewing window. Thus, the network would be capable of only extracting local drainage points. It would not be possible for the network to determine how a local drainage point might be connected to the global drainage network. This is a limitation which would prove to be fatal. A global, as well as a local view of the data is therefore necessary to determine whether a pixel is part of the overall drainage network.

How can we get a neural network to look at problem both locally and globally? Let's look at the human visual system to gain some insight into this problem. Our visual system is a perfect example of how many localized neurons can work together to give a global view of a data set. For our visual system the data set is light which is incident on the retina. The retina has several processing layers of neurons; in each layer neurons communicate with each other in an attempt to pick out features in the visual environment, such as lines at various orientations, etc. After several layers of processing signals are sent to the brain which represent a very non-local representation of the data. The brain receives not a direct mapping of the colors and intensities which strike the retina, but a code of prominent features in the visual field which through our evolution has been optimized. An individual neuron has a very localized scope, however, with thousands of localized neurons *communicating*, the global picture emerges. It is with our visual system in mind that we will attempt to gain a global view of a drainage network using neural sheets made of many locally connected neurons.

It is important to understand the significance of using several processing sheets (layers) for a problem such as drainage network extraction. Figure 1 shows an example of a two layer system. The two large rectangles represent neural processing layers of arbitrary dimensions. These processing layers consist of many neurons (shown as circles) which interact with each other to carry out the desired data transformation. The first processing layer, in this case, is given a DTED matrix as input, which then feeds its output to the second processing layer which generates the final output. In general, each processing layer receives an input and transforms it into an output which may be fed to the input of another layer or used as the final output of the system. The transformation should be one which either simplifies the problem for the next layer

or produces a final output. A layer may simplify a problem in many ways. For example, it could enhance important features, reduce noise, or simply reduce the number of inputs into the next layer. Hence, we have a modular design with each layer simplifying the problem until an acceptable solution is reached. This research deals with designing the initial two processing layers for Dnet. These layers will perform the bulk of the drainage network extraction. If it is deemed necessary, other layers may be included in the system to make up for deficiencies in the DTED files and enhance the output of the initial layers.

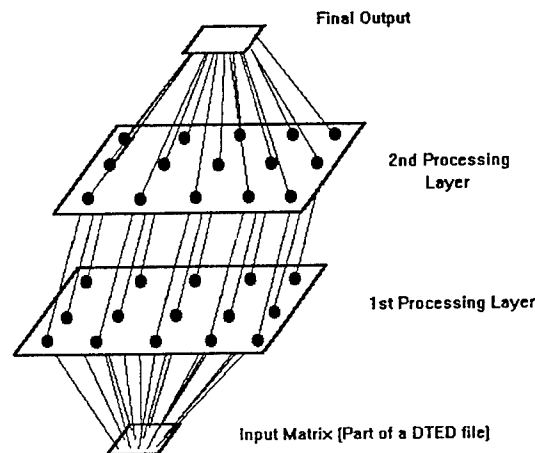


Figure 1. A Two Layer Dnet Processing System.

To design the network it is important to visualize the problem. Picture a hilly landscape with drainage valleys leading to large basins. Now dump water on this landscape and picture it's flow along the topology. Generally, water will flow downhill until drainage valleys are reached and continue to flow toward the large drainage basins. Now place a sheet of neurons over the topology so that each neuron corresponds to an (x,y) coordinate of the region under consideration. The neural sheet is arranged so that all neurons are connected via weights to their immediate neighbors. Assign to each neuron the elevation value at it's coordinate and also give each neuron an activity level of finite range. If the activity levels of the neurons could be made to correspond to the flow of water across the topology, a solution would be present. The goal is to have neurons which correspond to drainage points become highly active while other neurons become relatively inactive. To do this the weights between neighboring neurons must act so as to guide the network through a *trajectory* until it reaches an acceptable solution. Below are several drainage properties along with a description of how these properties relate to the neuron interactions. These properties will be the backbone of the network design and will be referenced in the Mathematical Model section as properties 1, 2, and 3.

- **Water flows from higher to lower elevations**⁽¹⁾. This corresponds to neurons at higher elevations exciting those below them, and neurons at lower elevations inhibiting those above them. *Excitation* of a neuron increases it's activity, while *inhibition* decreases the activity.

- **If an area is relatively flat, it will either be all drainage or all non-drainage**⁽²⁾. This is because water flowing into a flat area will distribute itself along the surface. If the area is a basin or a flat channel the water will accumulate and correspond to drainage. If the area is at a higher elevations relative to the rest of the topology, the water will run off the edges and correspond to non-drainage. How can a neuron in the midst of a flat area which can only see it's immediate neighbors know whether it is in a basin or not? This can be accomplished by simply *following the leader*. To understand this better, lets suppose that the activity of a neuron in a flat area is inclined to change by an amount proportional to the activity change of each of it's neighbors, hence following the trajectory of it's neighbors. Neurons in the middle of a basin do not know whether they are at a high or low elevation compared to the topology as a whole. Those at the edges of the flat area, however, will be excited by the neurons surrounding the basin due to property 1. Because of this positive activity change, neighbors of the edge neurons will also be excited and so will their neighbors. This *activity front* will collapse on the flat region until the center most neurons are reached.
- **Drainage points must be connected to other drainage point**⁽³⁾. Otherwise we would be considering local minima. This corresponds to having a decay term for all neurons which is counteracted only if a neuron borders other drainage points.

It is not being claimed that these properties alone characterize a fully connected drainage network. These simple properties, however, will allow for a single neural layer to make a very good guess as to what represents drainage regions and what does not.

2.2 Mathematical Model

To create the mathematical model for Dnet the following steps will be taken. First, constraints will be placed on the network model which will limit the complexity of the parts from which the network is built. Next, a method of indexing the variables associated with each neuron will be discussed. Using the described indexing techniques, the three drainage properties will be expressed in numerical terms. The mathematical implementation of the three drainage properties will be put into neural network terminology, completing the mathematical model of the first layer. Finally, a description of the second processing layer also known as the thinning layer will be given.

2.2.1 Constraints

To be assured that the network is designed according to the premise that it be composed of many simple parts and can be implemented in hardware, the following constraints have been placed on it's construction. These constraints place limits on the operational complexity and knowledge each component of the network may have.

- Each neuron may only have knowledge of a small number of internal variables. They also have the capacity to connect to a finite number of weights. They may not have direct access to other neurons.

- The job of a weight is to transform the activity of a source neuron to a useful mapping of that activity, which is then seen by a destination neuron. It may only possess knowledge of the source and destination neurons.
- The computations performed by the neurons and weights should be *simple* ones, ideally one step operations. They should not carry out multiple step algorithms.
- The network will be arranged in a sheet like structure with one neuron for every point of elevation data. Each neuron will be connected via a weight to it's neighboring neurons, most of which have 8 neighbors except for those on the region boundaries, which have either 3 or 5 neighbors.

This system will have only two neural sheet at the present time. If it is necessary, we will be free to add other layers to the system.

2.2.2 Indexing

The neuron variables can be indexed in two ways; it will become apparent that this dual indexing technique simplifies the notation. *Direct indexing* is shown below:

- E_p the elevation of neuron p .
 $Y_p(t)$ the activation of neuron p at time t ; in the interval from 0 to 1.
 p the (x,y) location of the neuron in the topology under consideration.
 t an integer time variable which is incremented by one for each network iteration.

And now for *neighborhood relative indexing*, we have:

- E_{pi} the elevation of the i 'th neighbor of neuron p .
 $Y_{pi}(t)$ activation of the i 'th neighbor of neuron p at time t ; in the interval from 0 to 1.
 i an indexing integer in the range of $[1 .. N_p]$.
 N_p number of neighbors surrounding neuron p .

Figure 2 shows neighborhood relative indexing of neuron activities. Each circle represents a locally connected neuron. The weights are shown as arrows between neighboring neurons. The arrows point from a source neuron to a destination neuron. There are two weights between every pair of neighbors in order to transmit information in both directions. Note that not all the weights are shown for clarity. In the real system each neuron is connected to every neighbor.

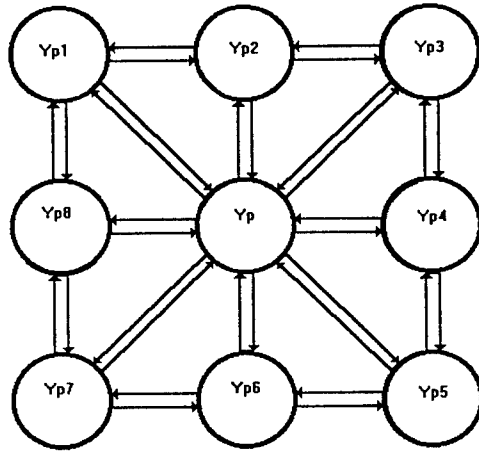


Figure 2. Example of Neighbor Connected Neurons Using Neighborhood Relative Indexing.

Using the variables and indexing techniques discussed, let's now implement the 3 drainage properties presented at the end of the *Conceptual Approach* section. For each property we will end up with a factor which when added to the activity of a neuron will bring us closer to the main objective. This is to have highly active neurons ($Y_p=1$) for drainage points and non-active neurons ($Y_p=0$) for non-drainage points.

2.2.3 Model for Property 1

To implement the first property a factor proportional to the product of two functions, as given in (2), will be added to activity of a destination neuron. The first function, as given in (1), operates on the elevation gradient between the destination neuron and its neighbor. The second function, $Y_{pi}(t)$, is the activity level of a particular neighbor. It is tempting to make the gradient function a linear mapping of the gradient, however, if an erroneous point exist whose elevation is much larger or smaller than the other neurons the stability of the system will be poor. To deal with such noise, a limit needs to be placed on the minimum and maximum gradients. Placing the gradients through a sigmoidal function would serve such a purpose. Since gradients may be either positive or negative the hyperbolic tangent function, which ranges from -1 to 1 is best suited for our application. This function is shown in Figure 3. By multiplying this function by a constant, S_{max} , the gradient measure is constrained between $-S_{max}$ and S_{max} .

$$S_{pi} = S_{max} \cdot \tanh(\alpha_{pi} \cdot (E_{pi} - E_p)) \quad (1)$$

$$\psi_{pi}^1(t) = S_{pi} \cdot Y_{pi}(t) \quad (2)$$

α_{pi} is a localized parameter used to adjust the rise time from $-S_{max}$ to S_{max} . To the activity of each neuron we will add $\psi_{pi}(t)$, which implements property 1 while allowing for noisy elevation values. The adjustable parameters for this property are S_{max} and α_{pi} .

2.2.4 Model for Property 2

The bell shaped function defined by (3) and shown in Figure 4 is an unnormalized Gaussian function with zero mean. It will be useful in the implementation of property 2.

$$\Phi(x, \sigma) = \exp\left(-\frac{x^2}{\sigma^2}\right) \quad (3)$$

This function is maximum at $x=0$, and smoothly decays to 0. This is a useful property since when it operates on gradients it's magnitude gives us a measure of the flatness of an area. Equation (4) is defined to give us a flatness measure between neighboring neurons. Where N_{\max} and σ_{pi} define an upper limit and localized variance of N_{pi} , respectively. To the activity of each neuron we will add the factor given in (5).

$$N_{pi} = N_{\max} \cdot \Phi((E_{pi} - E_p), \sigma_{pi}) \quad (4)$$

$$\Delta Y_{pi}(t) = Y_{pi}(t) - Y_{pi}(t-1)$$

$$\psi_{pi}^2(t) = N_{pi} \cdot \Delta Y_{pi}(t) \quad (5)$$

This will cause for the activity of a neuron to be changed in the same direction as neurons with approximately the same elevation value, implementing property 2's *follow the leader* characteristic. The adjustable parameters for this property are N_{\max} and σ_{pi} .

2.2.5 Model for Property 3

To implement property 3 we will begin by defining a decay term D which is a positive constant subtracted from the activity of each neuron during each iteration, as seen in (12). This will cause for an unconditional decrease in the activity of the neurons. Those which have other drainage neurons as neighbors will wish to counteract this decay term. To do this, we need a measure to indicate the likelihood of a neuron's neighbor belonging to a drainage area. This measure will be referred to as a connectivity measure and denoted by the variable $C_{pi}(t)$, defined by (6a-f).

$$C_{pi}(0) = 0 \quad (6a)$$

$$\text{IF } \left((Y_p(t) - Y_p(t-1)) > 0 \right) \text{ AND } \left((Y_{pi}(t) - Y_{pi}(t-1)) > 0 \right) \quad (6b)$$

$$\text{THEN } C_{pi}(t+1) = C_{pi}(t) + \Delta C \quad (6c)$$

$$\text{ELSE } C_{pi}(t+1) = C_{pi}(t) \quad (6d)$$

$$\text{IF } (C_{pi}(t) > C_{\max}) \quad (6e)$$

$$\text{THEN } C_{pi}(t) = C_{\max} \quad (6f)$$

The above conditions indicate that $C_{pi}(t)$ will increase as the activity of both the source neuron and its neighbors increase. The second IF/THEN condition simply places a threshold on the value of $C_{pi}(t)$. This measure will be optimized as our experience with the network increases. As a verification that connectivity between two neurons does exist, $C_{pi}(t)$ will be multiplied by a function of the neighboring neurons activity. Equation (7) has such a characteristic, increasing from 0 to 1 as a neuron approaches saturation. Finally, (8) is added to the activity of a neuron to account for property 3.

$$\vartheta(x, \beta) = \frac{1}{1 - \beta} \cdot (x - \beta) \cdot [u(x - \beta) - u(x - 1)] + u(t - 1) \quad (7)$$

$u(x)$ = unit step function

$$\psi_{pi}^3 = C_{pi}(t) \cdot \vartheta(Y_{pi}(t), \beta) \quad (8)$$

In the above equations, β is a value between 0 and 1 which defines when the function begins to rise. The adjustable parameters for this property are β , C_{max} , and ΔC .

2.2.6 Dnet Model

To arrive at our model, we use two layers of processing as shown in Figure 1. The first layer will take care of the bulk of the processing by implementing all three drainage properties. The second layer is optional, and is concerned with thinning rivers and streams. The second layer also helps remedy clustering, which is when non-drainage points bordering a drainage region are mistakenly labeled as drainage points by the first layer. A description of layer one will be given followed by a rather trivial description of the second processing layer.

2.2.6.1 First Processing Layer

Each weight will be a three dimensional vector as opposed to traditional networks which have one dimensional weights. The advantage of having multi-dimensional weights is that each component can represent a weighting of a particular characteristic of the source neuron's activity. We define the three weight vector components to be (1), (4), and (6), as given in (9).

$$\vec{W}_{pi}(t) = [S_{pi}, N_{pi}, C_{pi}(t)] \quad (9)$$

Each Dnet weight vector operates on an activity vector whose components represent a characteristic of the source neuron's activity. The activity vector is defined as the second factors of (2), (5), and (8), as given in (10).

$$\vec{A}_{pi}(t) = [Y_{pi}(t), \Delta Y_{pi}(t), \vartheta(Y_{pi}(t))] \quad (10)$$

The total input into a destination neuron may now be expressed as the summation of the inner products of the weight vectors with the corresponding activity vectors. This is formally expressed in (11).

$$X_p(t) = \sum_{i=1}^{N_p} \tilde{W}_{pi}(t) \cdot \tilde{A}_{pi}(t) \quad (11)$$

The network is updated in an iterative manner by cycling through each neuron and changing their activity levels according to (12).

$$\begin{aligned} Y_p(t+1) &= \vartheta [(Y_p(t) + X_p(t) - D), 0] \\ Y_p(0) &= Y_0 \end{aligned} \quad (12)$$

Where Y_0 is just an initial value between 0 and 1, and $\vartheta(.,.)$ is being used as the non-linear activation function of the neurons. The network should be iterated until the majority of the neurons approach either 0 or 1. A safe stopping condition is to iterate the network until 90 percent of the activities are either below 0.1 (signifying non-drainage) or above 0.9 (signifying drainage).

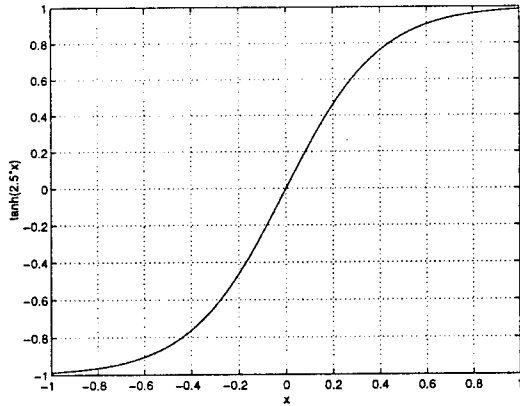


Figure 3. Plot of $\tanh(x)$.

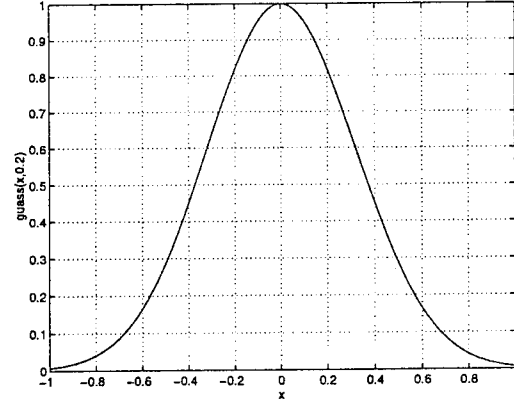


Figure 4. Plot of $\Phi(x, \sigma)$.

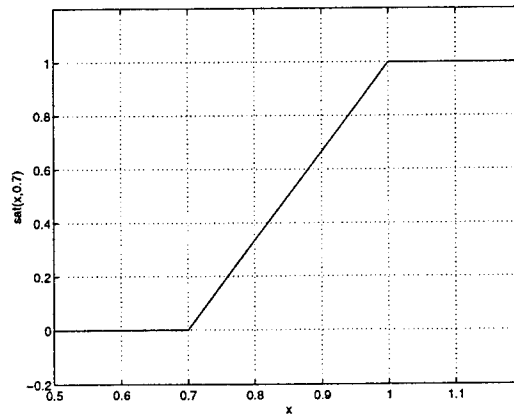


Figure 5. Plot of $\vartheta(x, \beta)$.

2.2.6.2 Second Processing Layer

After the first layer is iterated, it's output is given to the second layer as an input. Many times the rivers and streams extracted by the first layer are more than one pixel in width. The reason for this becomes apparent when the rivers and streams are viewed as valleys. It is difficult, if not impossible, to determine how far up the valley wall the water level of a particular river reaches, using only elevation data. Many geological factors which are independent of elevation help determine the water level in a particular valley. Figure 6a and Figure 6b each show a cross section of the same valley. The water levels, however, are not the same due to elevation invariant factors. Because of this the river widths $d1$ and $d2$ are different, showing the difficulty in determining the river width using only DTED.

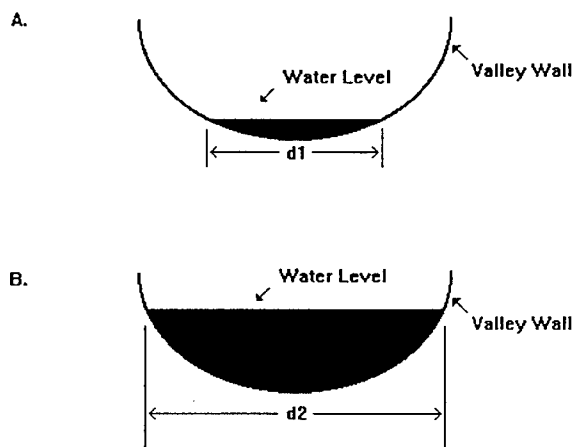


Figure 6. Example of Difficulty in Determining River Width.

Tribe's [5] approach to this problem is to first extract a drainage network with a width of one cell. Then using the gradients between the extracted network and the pixels adjacent to it, an adjacent pixel is labeled as being part of the drainage network if it's gradient is below a specified threshold. Dnet's approach to this problem is to guess the width of each stream segment during the first layer of processing. Recognizing that some application may only require a one pixel representation of stream segments, the second layer of Dnet transforms the first layer's output into such a representation.

The second layer (thinning layer) has the same dimensions as the first layer. The activities of the first layer are rounded to either zero or one and are copied to the corresponding neurons in the thinning layer. The same variables and indexing techniques used for the first layer will be used to implement the thinning layer. The following rule will implement the thinning process:

IF $(Y_p \neq 0.0)$ AND $(H_p \geq H_{cut})$ (Y_p has been rounded)
 THEN $Y_p = 0.0$

$$H_p = \sum_{i=1}^{N_p} \text{sgn}(E_p - E_{pi})$$

$$\begin{aligned} \text{sgn}(x) &= 0, x \leq 0 \\ &= 1, x > 0 \end{aligned}$$

Essentially the rule causes any node which has been labeled as drainage by the first layer and is at higher elevation than H_{cut} neighbors or more to be labeled as non-drainage. With the proper choice of H_{cut} , drainage points which are not at the minimum of a valley cross section will be labeled as non-drainage. To define the thinning layer in neural network terms a weight vector between neighboring neurons is given as (13). Equation (14) provides the total input to the neuron and (15) gives the update rule to find the new activation level.

$$W_{pi} = [\text{sgn}(E_p - E_{pi})] \quad (13)$$

$$X_p = \sum_{i=1}^{N_p} W_{pi} \quad (14)$$

$$Y_p = \text{sgn}(Y_p) \cdot \text{sgn}(H_{cut} - X_p) \quad (15)$$

Notice that there is no time variable in equations 13-14. This is because the second layer requires only 1 iteration after it receives the first layers output. The adjustable parameter for the second layer is H_{cut}

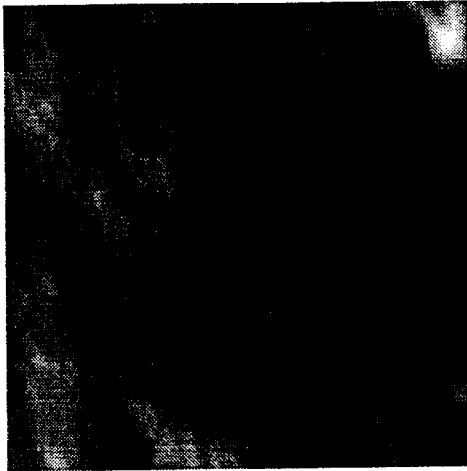
3. Experiments

This section will describe our results after experimenting with Dnet. It is followed by a discussion of the effects of each network parameter. Dnet was implemented in ANSI-C and was tested on a DEC Alpha station. For the longest trial, the network took no longer than 4 minutes.

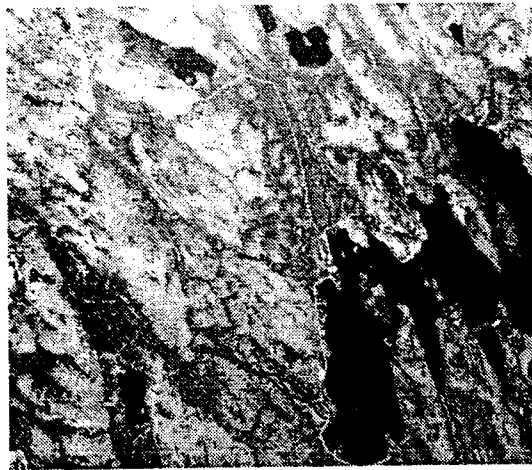
3.1 Testing Data

Two testing regions were selected from a 1 degree by 1 degree DTED cell (1201x1201 resolution) whose upper left corner is located at latitude 43N and longitude 75W, these regions will be referred to as test regions 1 and 2. Figure 7 shows a grey level representation of the DTED, an aerial image, and a topographical map of test region 1 which is of Raquette lake (the larger lake) in New York. Note that in 7a the elevation values were normalized from 0 to 255 and displayed as a grey level image, darker pixels correspond to lower elevations. The DTED region is 150 x 150 pixels. Figure 7b and Figure 7c were used as reference images during network testing. Note that the aerial image had a resolution of 750x1000 (rowsxcolumns).

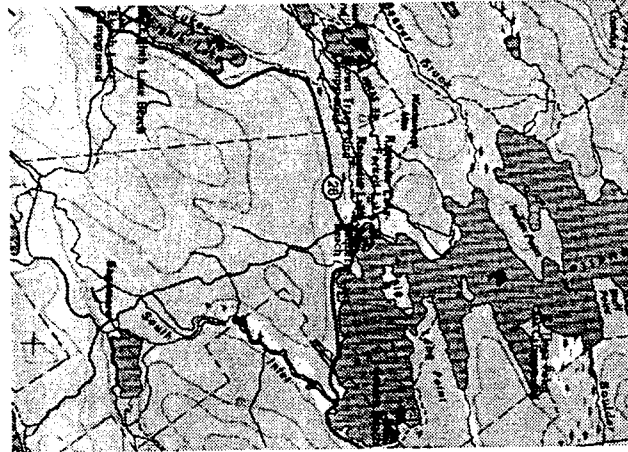
Figure 8 shows test region 2. This region is within the Pico and Panther mountains in New York and represents more of a challenge to Dnet since the terrain is so rough. It is more likely that a rough terrain will house false valleys and ridges which increases the difficulty associated with drainage network extraction. An aerial image of this region was not available to us; therefore, we used the topographical map of the area for evaluation.



(a) Grey level DTED.



(b) Aerial Image.



(c) Cartographic Map.

Figure 7. Test Region 1.



(a) Grey Level DTED



(b) Topographical Map

Figure 8. Test Region 2.

3.2 Results

In this section the network will be tested on regions 1 and 2. For each region the evolution of the network will be shown and described until the final result is reached. Finally the network parameters described in the *Mathematical Model* section will be discussed, explaining the effect that each parameter has on the network.

3.2.1 Test Region 1

A 150 by 150 node network was created and the DTED values of Figure 7a were given to it as input. Figure 9b is a grey level depiction of the network's output after only 2 iterations. A darker grey level corresponds to a smaller neuron activity at a particular pixel. At this early stage of the network's evolution the outline of the lakes and several possible river segments are already becoming apparent. This is due entirely to the sigmoidal component of the first layer's weight vector. Notice that the lakes are not filled in. This is because the normal component of the first layer's weight vector, which implements the *follow the leader* effect, has not yet had a noticeable impact.

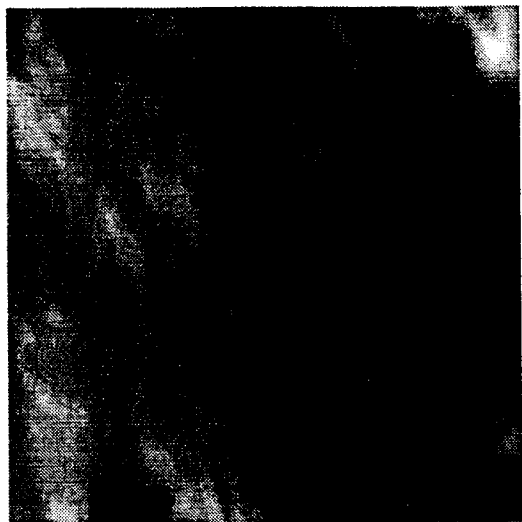
Figure 9c shows the network's activities after 30 iterations. The Gaussian component of property 2 is now having an effect. Notice that the smaller lakes have been completely filled in and the largest lake is in the process of being filled in. After 250 iterations, Figure 9d shows that the network has successfully extracted the lakes from the DTED and has selected many possible stream segments. At this point several stream segments are discontinuous since the connectivity factor has not yet had time to come into play. Figures 9e and 9f show the final output of the first and second layers respectively after 750 iterations. It is obvious when comparing these two images why the second layer is referred to as the *thinning layer*, as it transformed wide stream segments seen in Figure 9e into the one pixel stream segments found in Figure 9f. It is also important to recognize the role that the connectivity factor played in producing the final output. The streams and rivers have become less fragmented. This is due to the connectivity factor canceling out the decay term near the end of the networks evolution.

The final output shows that the network does an excellent job at extracting lakes. It also does a good job at extracting rivers and streams. It is more difficult to distinguish rivers and streams, partly due to the low resolution of the DTED. This causes for narrow rivers and streams to be poorly represented by the DTED. Many rivers and streams vary in their width, at narrow sections the river may seem to disappear when viewed from a DTED perspective. Another problem noticed was that for several cases the contour lines of a particular region appeared to be valley segments when viewed from a DTED perspective. Most of what the network extracted does correspond to drainage regions. There are problems mainly with stream and river connectivity and the inclusion of artificial valley segments. These are problems which could best be dealt with by adding another layer to the network.

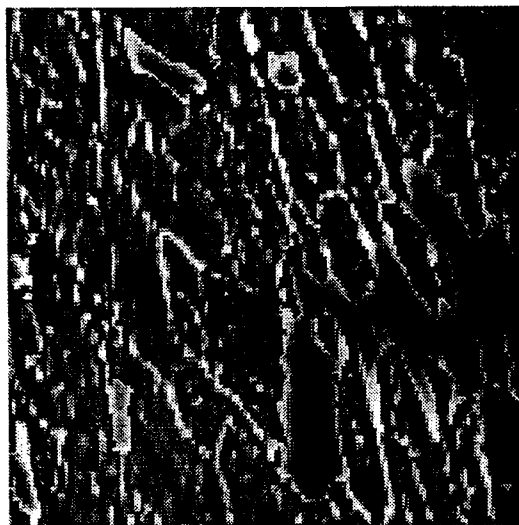
3.2.2 Test Region 2

Again a 150 by 150 node network was used for test region 2 with the DTED shown in Figure 8a being used as input. Use Figure 8b as a reference for test region 2.

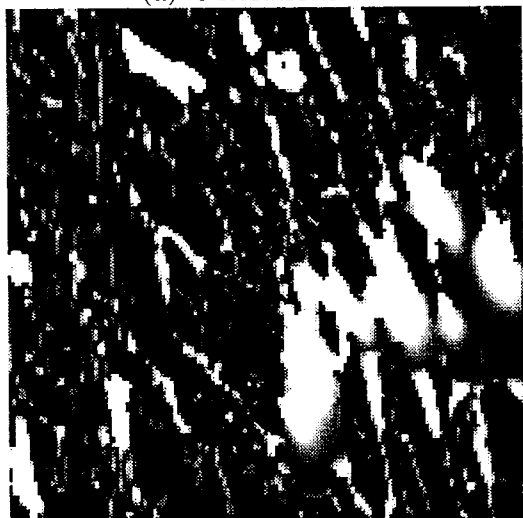
Figure 10b is a grey level depiction of the network's output after only 2 iterations and Figures 10c-f show the evolution of the network as described for test region 1. Note that this region represents a very rough terrain, thus the problems associated with the low resolution of the DTED will be amplified. The network was able to extract the lakes without any problem. The rivers and streams, however, are fragmented and some artificial valleys were labeled as drainage. By carefully comparing the network's output to Figure 8b it is obvious that the network was doing a good job. Most of what the network extracted does actually correspond to the drainage regions seen in the Figure 8b. The network's output is fragmented and includes some artificial valley segments, but it is good enough so that with the addition of another processing layer even better results should follow.



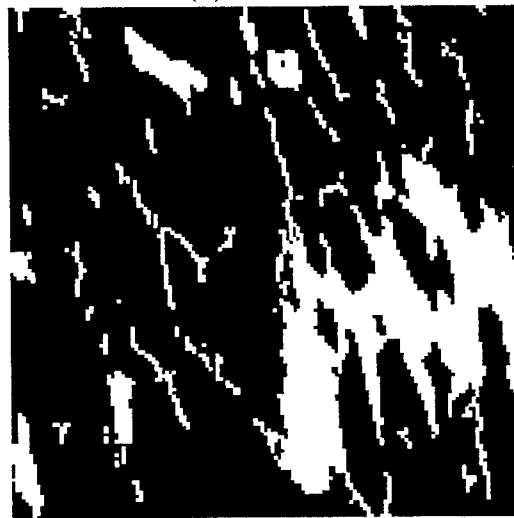
(a) 0 iterations.



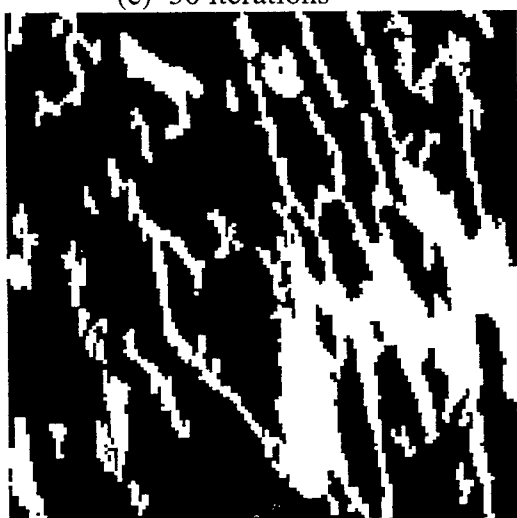
(b) 2 iterations.



(c) 30 iterations



(d) 250 iterations.



(e) 750 iterations (layer 1)

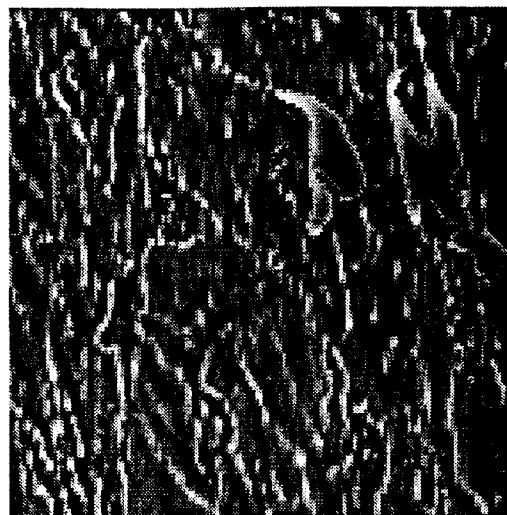


(f) 750 iterations (layer 2)

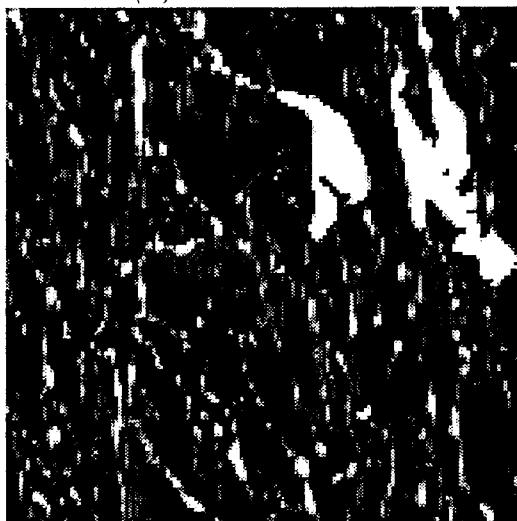
Figure 9. Test Results from Region 1.



(a) iterations.



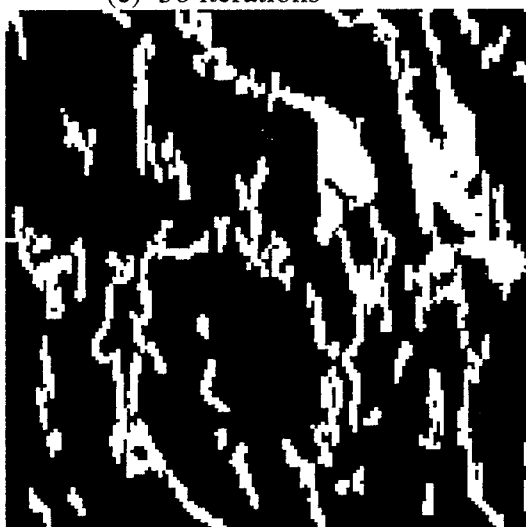
(b) 2 iterations.



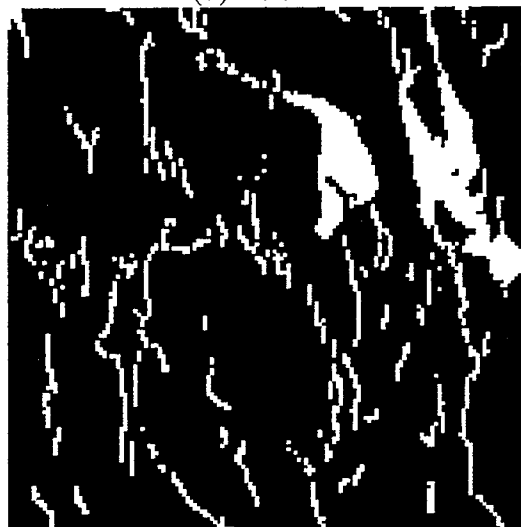
(c) 30 iterations



(d) 250 iterations.



(e) 750 iterations (layer 1)



(f) 750 iterations (layer 2)

Figure 10. Test Results from Region 2.

3.3 Discussion of Parameters

Here the effects of each parameter will be discussed and the values used for the testing results will be given.

3.3.1 S_{\max}

This parameter to a large degree controls how fast the network moves through its trajectory. If it is too large the neurons will saturate before the connectivity factor has a chance to increase. This will cause for drainage neurons to decay during the end stages of the network's evolution since the connectivity factor is not large enough to cancel out the decay term. If the value of S_{\max} is too small the network will take much too long to finish iterating. The value used for the testing results was:

$$S_{\max} = 0.05.$$

3.3.2 α_{pi}

This parameter controls how sharply the *tanh* function rises from -1 to 1. For larger values of α_{pi} the transition is more abrupt. The reason for using the non-linear *tanh* function was to lessen the effects of outliers. It does this by saturating at the extremes of its range. α_{pi} can be used to control where saturation occurs. If the function is to saturate at $x = -S_{\text{sat}}$ or S_{sat} then a safe value for the scale factor is:

$$\alpha_{pi} = \frac{1.8}{S_{\text{sat}}}.$$

Practice has shown that a good localized saturation point (S_{sat}) is 2 times the mean of the elevation gradients between a neuron at E_p and its neighbors.

$$\alpha_{pi} = \frac{1.8}{\frac{2}{N_p} \cdot \sum_{i=1}^{N_p} |E_p - E_{pi}|}.$$

3.3.3 N_{\max}

If this parameter is too small, the normal weight component will not have enough influence to do its job, which again is to label relatively flat areas as either drainage or non-drainage. If it is too large, the normal weight component may influence neurons other than those in flat areas, which would increase the error. A good value was found to be:

$$N_{\max} = 0.2.$$

3.3.4 σ_{pi}

This parameter controls how sharply the Gaussian function approaches its limits of zero on each side of its mean. Since our mean is zero, a value of σ_{pi} which will cause saturation at $x = -N_{sat}$ or N_{sat} is:

$$\sigma_{pi} = \frac{N_{sat}}{\sqrt{2}}.$$

Practice has shown that a good saturation value (N_{sat}) is 1.5 times the mean of the gradients between a neuron at E_p and it's neighbors.

$$\sigma_{pi} = \frac{\frac{1.5}{N_p} \cdot \sum_{i=1}^{N_p} |E_p - E_{pi}|}{\sqrt{2}}.$$

3.3.5 ΔC

If this parameter is too small connectivity of the final output will be poor, since the connectivity weight component will not reach a large enough value. On the other hand if it is too large the connectivity factor of non-drainage neurons may become too large and they will be erroneously labeled as drainage neurons. A good value is:

$$\Delta C = 0.025.$$

3.3.6 C_{max}

This is a critical parameter, since if it is not large enough it cannot perform it's job. The value of this must be larger than the sum of any negative input into a neuron. Since the only negative input may come from the decay term and the sigmoidal weight component, a safe condition is as follows:

$$C_{max,pi} > |S_{pi}| + D$$

Practice has shown a good choice for $C_{max,pi}$ is:

$$C_{max,pi} = |S_{pi}| + D + 0.004$$

3.3.7 β

This is not an overly critical parameter. It just needs to be reasonably close to the saturation value of the neurons. A value that worked well was:

$$\beta = 0.7.$$

3.3.8 D

This parameter should be relatively small when compared to S_{\max} and N_{\max} . It's purpose is only to add a small negative component to the activity of a neuron after the effects of the sigmoidal and normal weight components have died off. A good value was found to be:

$$D = 0.006$$

3.3.9 Y_0

Increasing this parameter is equivalent to dumping more water on the landscape. If this parameter is not large enough, most neurons will immediately decay to zero. A good value for this parameter is:

$$Y_0 = 0.24$$

3.3.10 H_{cut}

A good choice for H_{cut} is 3, however, for data which is extremely noisy it maybe necessary to increase this value, thus:

$$H_{\text{cut}} = 3.0$$

4. Conclusions

A biologically inspired neural network architecture for the automatic extraction of drainage networks from elevation data has been presented. By using two layers of locally connected neurons, it was shown that the network was able to distinguish between drainage and non-drainage pixels. Problems were encountered with the detection of false valleys and ridges, which can be largely attributed to the low resolution of the elevation data used for testing. The network's performance will improve as the resolution of the data is increased. The described architecture also allows for additional processing layers to be easily added to the network. Such extensions would increase the network's accuracy, particularly on low resolution elevation data. This report has demonstrated that neural networks are effective tools and should be considered when solving other problems in the geosciences.

5. Implementation

The entire DNet program was implemented in ANSI-C and compiled and tested on a DEC Alpha workstation running the UNIX operating system. To compile the program it is necessary to include the standard math library, the following command-line can be used on most any UNIX station:

```
cc dnet.c -lm -o dnet
```

This will result in an executable file named, **dnet**, which is ready to run. Running the program with no command-line arguments results in the usage information being displayed to the screen.

```
Usage: dnet Input Output Height Width Iter SaveInt  
[-d (display)] [-t (thin)] [-w (weights)] [-g (grey)] [-b (bin)] [-o (org)]
```

The first six arguments are required to run the dnet program. The last six arguments are optional and can be specified in any order. Below is a discussion of each of the command-line arguments.

5.1 Input

This parameter specifies the filename of the DTED image to be used as the input to DNet. The input image should be an ASCII image taken from a section of a DTED cell. We have written a program called, **dted2asc.c**, which accepts the name of a DMA formatted 1201x1201 DTED cell as input and generates an ASCII version of the cell. The command-line, assuming that **dted2asc.c** has been compiled, for this conversion program is:

```
dted2asc <dted file name> <ASCII output filename>
```

The program generates a 1201x1201 ASCII image which can be viewed with an image processing software such as Khoros. This image is too big to be given to DNet as input. First the image must be divided into sub-regions. We generally used images which had dimensions of 300x300 or less. The generation of the sub-regions can be accomplished with most any image processing software.

5.2 Output

The name of the output file to be generated by the DNet program. The program adds an iteration number and an extension to this name in order to generate the various output files names. The extensions are specified by the optional arguments which will be discuss later.

5.3 Height

The number of rows in the input image.

5.4 Width

The number of columns in the input image.

5.5 Iter

The number of **iterations** for which DNet is to be run.

5.6 SaveInt

This is the **save interval**. After every 'SaveInt' iterations an output image will be generated which uses 'Output', the iteration number and an added extension for the filename. If SaveInt = 0 then only the final result will be saved.

5.7 Optional Arguments

These next three arguments are optional and can be specified in any order.

- d : This option allows the network activity levels to be **displayed** to the screen while it is iterating. This option is only used for debugging with very small input files.
- t : This option will cause for the network output to be **thinned**, as described in the report.
- w : This will generate a data file containing the **weights** of every neuron in the network. Again this option is only used for debugging with very small input files. The data file will have the name of '**output**' with the added extension of '**.wts**'.

At least one of the next three arguments should be specified since they dictate what type of the output image the network will generate. If none are specified there will be no output. If more than one is specified there will be multiple outputs. The output images are ASCII with the same dimensions as the input.

- g : This specifies that the output image is to be a **grey level** representation of the activities of each neuron. The image file will have the name of '**output_iteration#**' with the added extension of '**.grey**'.
- b : This specifies that the output is to be a **binary** representation of the network activities. The threshold of binarization can be found in the define section of the program code and can be changed to any value between zero and one. The image file will have the name of '**output_iterations#**' with the added extension of '**.bin**'.
- o : This specifies that the output is to be a binary image of the output as described above multiplied by the **original** input. Thus the resulting image has the original

elevation values at each non-zero pixel in the binary image. The image file will have the name of **'output'** with the added extension of **' .org'**.

The parameters discussed in the report are specified at the beginning of the program in the define section. The comments explain how the program parameters relate to the parameters discussed in the report.

6. References

-
- [1] Lee, J., P. K. Snyder, and P. F. Fisher, "Modeling the Effect of Data Errors on Feature Extraction from Digital Elevation Models," *Photogrammetric Engineering & Remote Sensing*, vol. 58, no. 10, Oct. 1992, pp. 1461-1467.
 - [2] O'Callaghan, J.F. and D. M. Mark, "The extraction of drainage networks from digital elevation data," *Computer. Vision, Graphics, and Image Process.*, vol. 28, pp. 323-344, 1984.
 - [3] Jenson, S.K., "Automated derivation of hydrologic basin characteristics from digital elevation model data," *U.S. Geol. Survey, Pub.*, Nov. 15, 1984.
 - [4] Qian, J., R. W. Ehrich and J. B. Campbell, "*DNESYS-An expert system for automatic extraction of drainage networks from digital elevation data*," *IEEE Transactions on Geoscience and Remote Sensing*, vol 28, no.1, pp. 29-44, 1990.
 - [5] Tribe, A., "*Towards the automated recognition of landform (valley heads) from digital elevation models*," *Proceedings of the 4th International Symposium on Spatial Data Handling*, 1:45-52, 1990

***MISSION
OF
ROME LABORATORY***

Mission. The mission of Rome Laboratory is to advance the science and technologies of command, control, communications and intelligence and to transition them into systems to meet customer needs. To achieve this, Rome Lab:

- a. Conducts vigorous research, development and test programs in all applicable technologies;
- b. Transitions technology to current and future systems to improve operational capability, readiness, and supportability;
- c. Provides a full range of technical support to Air Force Materiel Command product centers and other Air Force organizations;
- d. Promotes transfer of technology to the private sector;
- e. Maintains leading edge technological expertise in the areas of surveillance, communications, command and control, intelligence, reliability science, electro-magnetic technology, photonics, signal processing, and computational science.

The thrust areas of technical competence include: Surveillance, Communications, Command and Control, Intelligence, Signal Processing, Computer Science and Technology, Electromagnetic Technology, Photonics and Reliability Sciences.